HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Django & Design Patterns

Arnar Már Brynjarsson
Fridtjof Peer Stoldt
Hafliði Stefánsson
Ingunn Káradóttir
Katrín Viktoría Hjartardóttir
Kristján Mar Svavarsson
Maciej Sierzputowski
Valgerður Ásgeirsdóttir
Ægir Máni Hauksson

HONN – FALL 2020

Gerardo Reynaga

Gunnar Jörgen Viggóson, TA

Table of Contents

1. Preface

In this short report we will document what design patterns Django implements as a part of our project in HONN at Reykjavik University, Fall of 2020.

2. Registry

Registry is an already implemented design pattern in Django. In the Django source code, they created a class called apps that has an overview of all installed apps within the system and their configurations. As stated in the documentation for the registry in the Django: “A registry that stores the configuration of installed applications. It also keeps track of models, e.g. to provide reverse relations”. How we implement this is by using settings.py in the main registry app to link all other apps together. We have a known object that all other objects within the system can access. The applications also talk to the database through the registry (settings.py), first they call upon registry and then they will be redirected elsewhere to pull the data they need. If a new app is created and not linked to the registry, it will not be accessible, and it won’t have access to any other apps.

Here is an example form the registry of how all configurations can be accessed:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'user.apps.UserConfig',
    'workout.apps.WorkoutsConfig',
    'wallet.apps.WalletConfig',
    'payment.apps.PaymentConfig',
]
```

3. Layer Supertype

Layer Supertype is a class that acts as the supertype for all types in its layer. It's common for many objects in a layer to have the same methods. You don't want to duplicate these methods throughout the system so you can move all of this behaviour into a common Layer Supertype.

Django implements this design pattern for us by having a modelbase. In that modelbase there is a Metaclass for all models in the layer, there we have a model for methods such as create, delete and update. We use that in the classes we create ourselves, that helps us keep our code clean and without repetitions.